Graph Clustering-based Ensemble Method for Handwritten Text Line Segmentation

Vasant Manohar, Shiv N. Vitaladevuni, Huaigu Cao, Rohit Prasad, and Prem Natarajan
Speech, Language, and Multimedia Business Unit
Raytheon BBN Technologies
Cambridge, MA 02138
{*vmanohar, svitalad, hcao, rprasad, pnataraj*}*@bbn.com*

***Abstract*—Handwritten text line segmentation on real-world data presents significant challenges that cannot be overcome by any single technique. Given the diversity of approaches and the recent advances in ensemble-based combination for pattern recognition problems, it is possible to improve the segmentation performance by combining the outputs from different line finding methods. In this paper, we propose a novel graph clustering-based approach to combine the output of an ensemble of text line segmentation algorithms. A weighted undirected graph is constructed with nodes corresponding to connected components and edge connecting pairs of connected components. Text line segmentation is then posed as the problem of minimum cost partitioning of the nodes in the graph such that each cluster corresponds to a unique line in the document image. Experimental results on a challenging Arabic field dataset using the ensemble method shows a relative gain of 18% in the F_1 score over the best individual method within the ensemble.**

***Keywords*-text line segmentation; handwriting; ensemble method; graph clustering;**

I. Introduction

Text line segmentation of handwritten documents is one of the most difficult problems in developing a reliable OCR system. Unlike machine printed text, handwriting presents unique challenges such as touching and overlapping components, irregularity in geometrical properties of the line, such as line width, height, etc. Although the performance of text line segmentation methods have improved significantly in recent years, the robustness of such systems on real-world field data still has a large scope for improvement. For instance, a system that reported 99.55% using the FM metric in *ICDAR 2009* text line segmentation competition [1] achieved 56.1% using the same metric on a field dataset that included low resolution, noisy Arabic handwritten and mixed-type documents. A sample page from this corpus is shown in Fig. 1.

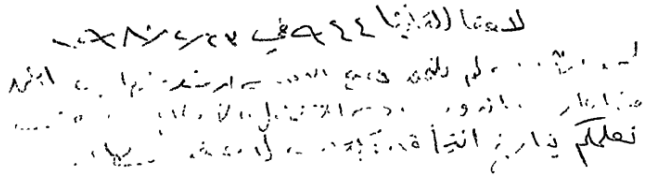

Fig. 1. Sample pre-processed document image from the field dataset used in this work showing additional challenges such as foreground fragmentation.

Text line segmentation methods can be broadly categorized into two classes: *top-down* and *bottom-up* methods. These include projection-based methods [2], smearing approaches [3], grouping-based techniques [4], Hough transform-based algorithms [5], and graph-based methods [6]. We refer the reader to [7] for a review of existing offline handwritten text line segmentation methods. Given the contrastive nature of the various methods and the recent success in applying ensemble-based combination for pattern recognition problems, it is possible to explore solutions that harness the complementary information within an ensemble framework.

Ensemble methods have been successfully applied in continuous speech recognition where, a system called ROVER (*Recognizer Output Voting Error Reduction*) [8] was developed to reduce the word error rate by aligning and combining the results from multiple speech recognizers. In the context of document recognition, a framework called StrCombo was presented in [9] for numeric string recognition where, a graph-based combination approach used each geometric segment of the individual recognizers as a node in a graph. The best path through this graph provided the final recognition result. In isolated word recognition, Wang et al. [10] proposed an approach where instead of using word classes, words were treated as sequences of character classes. A combination framework was presented which used a weighted opinion pool. Many researchers have successfully ported the ideas proposed in ROVER to the text recognition problem. Bertolami and Bunke [11] addressed the problem of text recognition using an ensemble of recognizers that were combined using ROVER. In [12], Prasad et al. developed a videotext recognition system that used ROVER for combining the hypotheses of a text region from multiple frames in the video. A survey of existing literature indicates that no study has proposed an ensemble approach for handwritten text line segmentation.

Two issues important to practical application of ensemble techniques are: (1) diversity among the individual methods that constitute the ensemble, and (2) an adequate combination strategy to exploit the results of different methods. In order to address the first aspect of the problem, we chose two top-down and two bottom-up approaches for our ensemble generation. For the second and more challenging facet of an ensemble method, we propose a data-driven combination strategy that constructs a co-occurrence graph with nodes

1520-5363/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDAR.2011.121

corresponding to connected components and edges connecting pairs of connected components with associated cost of putting the pair in the same line. The edge cost is determined by (1) the cost for a false split and false merger, and (2) the likelihood that a pair of connected components belong to the same line conditioned on the ensemble output; these likelihoods are learned during training. Text line segmentation is then composed as the problem of minimum cost partitioning of the nodes in the graph such that each cluster would correspond to a unique line in the document image. A key benefit of this method is that the number of clusters that denote text lines need not be specified a priori.

The principle contribution of this work is: (1) the formulation of an ensemble system as a graph clustering problem and (2) the application of the method for combining the output of multiple handwritten text line segmentation algorithms.

II. Methods for text line segmentation

In this section, we give a brief description of each line finding algorithm that constitutes the ensemble in this work.

A. Top-down methods

(I) Piecewise projection profile based approach [2]: In this method, lines were segmented based on piece-wise horizontal projection profiles of the document obtained at an interval determined by the average width of connected components and the page width. Once the projection profiles were obtained, initial set of candidate lines were generated by connecting the valleys in the current profile to the closest valley in the previous profile. For the unmapped valleys, a straight line was continued from the valley. Using the initial set of candidate lines, lines were drawn parallel. Any line drawn may be obstructed by a handwritten component. A decision was made to associate this component to the line above or below through a Gaussian probability decision based on the spatial proximity of the foreground pixels to the line.

(II) Directional filter based approach [13]: This method is based on *steerable directional* filter, which found the local orientation of a text line by scanning in multiple directions. The maximum response from a convolution of the filter with the image was the estimated direction of the text line. Specifically, the algorithm had the following key steps: first, a stroke segment that crossed a text line was automatically detected. Next, a reference line for splitting touching lines was estimated based on centers of gravity of the contours from the detected lines. Finally, touching components were split at the contour level and the character images were reconstructed.

B. Bottom-up methods

(III) Method based on filter banks and graph segmentation [6]: The first stage of the algorithm applied a bank of anisotropic Gaussian filters of different orientations and scales. The second stage modeled the document as an undirected weighted graph, where each connected component was represented by a node in the graph. Affinity Propagation (AP) method was then used to segment the graph. The advantage of using AP is that the number of sub-graphs that denote text lines need not be specified a priori.

(IV) Method based on baseline detection: In the first step of this method, any small dots or diacritics like components were removed from the input image. Then, baseline detection was performed by computing candidate lines that passed through text characters and the line that picked the most number of text pixels on its way was chosen. All connected components that passed through this line were marked so that in the next iteration these components were not included into the voting process. In order to prevent detection of false baselines, a dynamic threshold was estimated for the vote, which depended on the average text line length in the document. Once all baselines were estimated, text characters which were still unmarked were associated to the closest baseline. In the last step, diacritics were linked to the closest text character.

III. Graph clustering-based ensemble method

In this section, we describe the approach in terms of the structure of the graph on the document images' connected components, the edge-costs for the graph from the ensemble algorithms, and clustering the nodes in the graph to obtain text lines.

A. Structure of the graph

We construct a weighted undirected graph with nodes corresponding to connected components in the document image. Text line segmentation is coined as the problem of clustering the nodes in the graph. Let the constructed graph be $\mathcal{G} = (V, E)$. The graph's vertices are $V = \{v_i\}_{i=1}^{n}$ where v_i corresponds to the ith connected component in the document image, and n is the number of connected components. Let the edges be $E = \{e_{i,j} | v_i, v_j \in V\}$, and the cost associated with edge $e_{i,j}$ be denoted by $w_{i,j}$. Cost of edges not present in the graph are by default 0.

The edges in graph, $\mathcal{G}$, are determined by the pixel overlap between the connected components in the image and the line segmentations computed by the ensemble of algorithms. To this end, we first construct an adjacency matrix on the connected components for each algorithm in the ensemble. Next, the resultant adjacency matrices are collapsed into one graph $\mathcal{G}$ for the clustering. Two nodes v_i and v_j in $\mathcal{G}$ are connected with an edge if at least one algorithm in the ensemble puts them in the same line, or if v_i and v_j have a common neighbor in lines of two different algorithms. The formal definition is given below.

Let A denote a line finding algorithm, i.e. one of the 4 previously mentioned. Let $\mathcal{L}_{\mathrm{A}}$ be the set of lines computed by algorithm A, $\mathcal{L}_{\mathrm{A}} = \{L | L \subset \Omega\}$, where Ω is the set of image pixels. Let $\mathcal{M}$ be the set of connected components in the image, with a one to one correspondence between the connected components and graph nodes, $\mathcal{M} = \{M_i | M_i \subset \Omega \wedge v_i \in V\}$.

For each algorithm A, a labelling, N_{A}, is defined on the connected components based on the pixel overlap between the

connected components and the lines computed by A:

$$N_{\mathrm{A}}(v_i) = \arg\max_{L \in \mathcal{L}_{\mathrm{A}}} \frac{|M_i \cap L|}{|M_i \cup L|} \quad (1)$$

The labelling of the connected components by an algorithm A induces an adjacency matrix E_{A} on the connected components

$$E_{\mathrm{A}}(j,i) = E_{\mathrm{A}}(i,j) = \begin{cases} 1 & : \quad N_{\mathrm{A}}(i) = N_{\mathrm{A}}(j) \\ 0 & : \quad \text{otherwise} \end{cases} \quad (2)$$

The edges of $\mathcal{G}$ are defined as

$$\begin{aligned} E \quad = \quad & \{e_{i,j} | \exists \mathrm{A} : E_{\mathrm{A}}(i,j) = 1\} \cup \\ & \{e_{i,j} | \exists \mathrm{A}, \mathrm{B}, k : E_{\mathrm{A}}(i,k) = E_{\mathrm{B}}(j,k) = 1\} \end{aligned} \quad (3)$$

B. Cost of the graph edges

The cost of edge $e_{i,j}$ is determined by:

- estimated likelihood that the two connected components v_i and v_j should indeed belong to the same line, referred to as $p_{i,j}$. The $p_{i,j}$'s depend upon the document image and are computed from the output of the ensemble of segmentation algorithms.
- cost of making false merger and false split errors. This is a parameter that can be used to trade-off between mergers and splits among the lines.

To estimate the likelihood of two connected components, v_i and v_j, to belong to the same line, let us define a feature vector of their grouping according to the ensemble algorithms. Formally,

$$\mathbf{x}_{i,j} = \langle E_{\mathrm{I}}(i,j), E_{\mathrm{II}}(i,j), E_{\mathrm{III}}(i,j), E_{\mathrm{IV}}(i,j) \rangle$$

For instance, if algorithms I and III put v_i and v_j in the same line and algorithms II and IV put them in distinct lines then $\mathbf{x}_{i,j} = \langle 1, 0, 1, 0 \rangle$.

The likelihood of v_i and v_j belonging to the same line, $p_{i,j}$, is determined by the likelihood conditioned on the ensemble feature vector

$$p_{i,j} = \sum_{\mathbf{y}} P(v_i \text{ and } v_j \text{ in same line} | \mathbf{x}_{i,j} = \mathbf{y}) P(\mathbf{x}_{i,j} = \mathbf{y})$$

For an ensemble of 4 algorithms, $\mathbf{y}$ can attain 2^4 possible values. We use the training data to learn the conditional likelihood of any pair of connected components to belong to the same line given the output of the ensemble algorithms:

$$P(u \text{ and } v \text{ in same line} | \mathbf{x}_{u,v} = \mathbf{y}) = \frac{\sharp\text{events}[\mathbf{x}_{u,v} = \mathbf{y} \wedge \text{Groundtruth } u \text{ and } v \text{ in same line}]}{\sharp\text{events}[\mathbf{x}_{u,v} = \mathbf{y}]} \quad (4)$$

We set $P(\mathbf{x}_{i,j} = \mathbf{y}) = 1$ for the output combination generated by the ensemble for v_i and v_j, and 0 for the rest of the possible combinations.

This data-driven approach has the following advantages:

- For novel datasets, it is difficult to predict the performance of individual line segmentation algorithms used in the ensemble. Observing the success rates on the training data helps tune the edges costs to peculiarities of the dataset.
- Learning the likelihoods on the combined output of the ensemble captures how different combinations of the ensemble algorithms work. For instance, *is it the case that when algorithms II and IV put v_i and v_j in same line, then they are highly likely to be correct?*

Our application has more than **40,000 text lines** in training data; this amount of data easily allows for learning $2^4 = 16$ combinations using eq.(4). However, if the number of algorithms in the ensemble is large, say > 10, then learning a joint likelihood on the entire ensemble would require large amounts of training data. In such cases, it is possible to learn the likelihoods conditioned on subsets of the ensemble algorithms. This is not addressed in this paper and will be explored in future work. In this work, the estimated $p_{i,j}$'s ranged from 0.01 for $\mathbf{x}_{i,j} = \langle 0, 0, 0, 0 \rangle$ to 0.86 for $\mathbf{x}_{i,j} = \langle 1, 1, 1, 1 \rangle$.

Let the pairwise costs of merge/split decisions be:

Automatic clustering puts v_i and v_j in	Ground-truth v_i and v_j in	
	distinct line	same line
same line	$\lambda_{0,0}$	$\lambda_{0,1}$
distinct line	$\lambda_{1,0}$	$\lambda_{1,1}$

Here, $\lambda_{1,0} > 0$ and $\lambda_{0,1} > 0$ are the cost of false splits and false mergers, respectively. Similarly, $\lambda_{0,0} < 0$ and $\lambda_{1,1} < 0$ are the benefit of correct splits and correct mergers, respectively. The estimated cost of putting connected components, v_i and v_j, in the same line is $p_{i,j}\lambda_{1,1} + (1 - p_{i,j})\lambda_{1,0}$. Similarly, the estimated cost of putting them in distinct lines is $p_{i,j}\lambda_{0,1} + (1 - p_{i,j})\lambda_{0,0}$. For simplicity, we set $\lambda_{0,1} = \lambda_{1,0} = -\lambda_{0,0} = -\lambda_{1,1} = \lambda$. This results in the estimated cost of putting nodes, v_i and v_j, in the same lines as $w_{i,j} = (1 - 2p_{i,j})\lambda$.

C. Clustering the graph nodes

The problem is to partition the nodes into $\mathcal{C} = \{C_k\}$ subsets so as to optimize the following:

$$\begin{aligned} \min : \quad & \sum_{C \in \mathcal{C}} \sum_{v_i, v_j \in C} w_{i,j} \\ \text{s.t.} : \quad & C_k \cap C_l = \phi \quad \forall C_k, C_l \in \mathcal{C} \bigwedge \bigcup_{C \in \mathcal{C}} C = V \end{aligned} \quad (5)$$

When the number of clusters, $|\mathcal{C}|$, is known then this can be viewed as a k-min cut problem, which has a polynomial time algorithm for non-negative weights and known k. In our application, the number of lines in the image (number of clusters) is not known a-priori. Moreover, it is important to have both "must-link" (negative cost) and "don't-link" (positive cost) constraints. Finding the minimal cut is NP-Hard if the weights can be negative or when the number of clusters is unknown, making it unsuitable for the problem.

In general, the above optimization is a Quadratic Semi-Assignment Problem (QSAP) [14], known to be NP-Hard. Charikar presented semi-definite programming (SDP) and linear programming (LP) relaxations to the cluster problem in [15]. The LP formulation has the advantage of naturally

handling positive and negative weights, and not requiring a-priori knowledge of number of clusters. Vitaladevuni and Basri adapted the LP relaxation to the problem of co-clustering image segments in [16]. In particular, they modified the LP relaxation for practical applications involving thousands of graph nodes, referred to as LP-reduced (LP-R). Experiments reported in [16] indicate that LP-R outperforms alternative such as thresholding the adjacency matrix, Normalized cuts [17], Normalized cuts with negative weights [18], and the SDP-based approach.

The LP-reduced (LP-R) formulation constructs a metric space of distances between the graph nodes induced by clustering. Let $d_{i,j}$ denote the distance between nodes v_i and v_j in the cluster space. If $d_{i,j} = 0$, they are put in the same cluster; if $d_{i,j} = 1$, they are put in distinct clusters. Thus, the set of distances between all pairs of nodes defines the clustering. Metric properties of positivity, symmetry, and triangular inequality are enforced through linear inequalities. The LP-R relaxation of the optimization in eq.(5) is:

$$\begin{aligned} \max: \quad & \sum_{i,j} w_{i,j} d_{i,j} \\ \text{s.t.}: \quad & 0 \leq d_{i,j} \leq 1 \ , \ d_{i,j} = d_{j,i} \ , \ d_{i,i} = 0 \\ & d_{i,j} \leq d_{i,k} + d_{k,j} \quad \forall e_{i,j}, e_{i,k} \text{ and } e_{k,j} \in E \end{aligned} \quad (6)$$

Ideally, we would like the distances computed as a solution to LP-R in eq.(6) to be binary. However, linear programs do not guarantee integral solutions unless the constraint matrix is Totally Unimodular (TUM). It can be shown that in general the constraint matrix in eq.(6) is not TUM [16]. Generating a binary solution from a given real-valued LP solution while maintaining optimality is NP-Hard. However, in practice, the LP solutions generated in our application were very sparse, with predominantly binary values. In all our experiments, a simple thresholding at 0.6 was used to convert real-valued solutions to binary values. This is along the lines of empirical observations in [16].

When the cost/benefit of mergers and splits is set at $\lambda_{0,1} = \lambda_{1,0} = -\lambda_{0,0} = -\lambda_{1,1} = \lambda$, the optimization function in eq.(6) becomes $\lambda \sum_{i,j}(1 - 2p_{i,j})d_{i,j}$. Thus, the λ parameter has no effect and is set to $\lambda = 1$. In practical handwriting recognition tasks, the cost parameters will be useful for trade-off between false splits and mergers, which is important for optimizing recognition performance. We will explore individually varying $\lambda_{1,0}$, $\lambda_{0,1}$, etc. and their effect on word error rates in future work.

IV. Experimental results

The experiments were conducted on field data which consisted of 2477 Arabic handwritten and mixed-type (machine print and handwritten) pages scanned at 200 dpi. The pages can be characterized by high levels of noise and foreground fragmentation making the segmentation task highly challenging. The dataset was split into training, validation, and testing sets that included **2077**, **200**, and **200 pages**, respectively. The test set has 3352 text lines in total.

Prior to text line segmentation, we cleaned the artifacts in the input image by first removing the background noise through basic morphological filters. In the next step, we detected rule lines in the image, used them to detect and correct the skew, and finally removed pixels belonging to ruled lines, while preserving those that belonged to the glyph element.

We evaluated the line segmentation algorithms using the protocol established in [1] that computes precision and recall metrics by finding one-to-one mapping between the truth lines and the system hypothesized lines. The mapping function was pixel-based where a minimum fraction of overlap was required in order to declare a *hit*. We used 0.9 as the threshold for the overlap ratio in our experiments. We also report the F_1 score which is the harmonic mean of the precision and the recall.

Table I shows the precision, the recall, and the F_1 scores for each of the individual algorithms and the ensemble method on the test set. We observe **20%** relative gain in precision and **9.4%** relative gain in recall when compared to the corresponding best numbers. When pitched against the single best system within the ensemble in terms of the F_1 score, we observe a relative gain of **26.3%**, **11.1%**, and **18.3%** in the precision, recall, and F_1 scores respectively.

TABLE I
TEXT LINE SEGMENTATION RESULTS ON FIELD DATA.

Method	Precision	Recall	F_1 score
Projection profile [2]	52%	61%	0.561
Steerable filter [13]	50%	64%	0.561
Graph segmentation [6]	60%	53%	0.563
Method based on baselines	57%	63%	0.599
Ensemble	**72%**	**70%**	**0.710**

Fig. 2 presents the output of each individual algorithm within the ensemble, the associated errors in their output, and the output of the ensemble method. We see that there are significant errors in the output of the constituent algorithms – projection profile method is predominantly fragmented, steerable filter technique is largely fused, and both the bottom-up approaches have a mix of fragmentation and merge errors. On the other hand, the ensemble method surmounts these errors by efficiently integrating the individual decisions.

We made the following observations by analyzing the results on the individual pages:

- The ranking of the individual member's performance considerably varies across the dataset. More importantly, a single method does not outperform the other methods consistently. This reiterates the need to apply an ensemble approach.
- If we treat the best output from an individual method for a page as the result of a *page-level* Oracle, the ensemble method either surpasses or equals the Oracle performance on 89% of the pages in the test set. On 11% of the pages, we noticed that the output of majority of ensemble members is degraded to an extent where the combination does not overcome the individual errors produced by them.

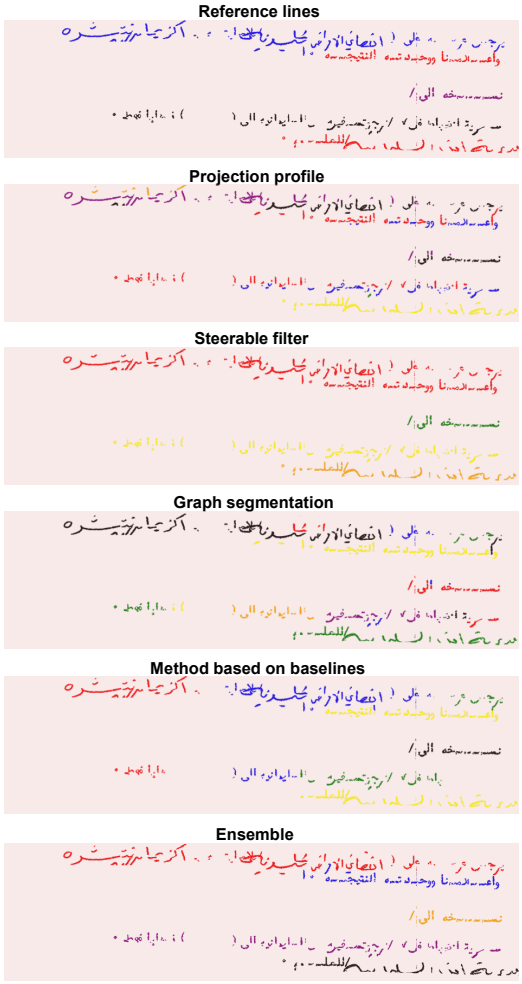

Fig. 2. Output of individual algorithms and the combination method on a sample page (text lines are color-coded for better illustration).

V. Conclusions and Future Work

In this paper, we have described a novel graph clustering based ensemble method for combining the output of multiple line segmentation algorithms. The method was applied to a large corpus of real-world Arabic handwritten and mixed-type document images and showed significant improvements in the precision and recall metrics when compared to the individual line finding methods. By allowing a soft combination of outputs using likelihood estimates, the framework provides the flexibility to adapt the confidence associated with an ensemble member for diverse datasets. Future work includes scaling to large ensembles and estimating the optimal trade-off between false splits and merges w.r.t. the word error rate.

Acknowledgments

This paper is based upon work supported by the DARPA MADCAT Program. The authors would like to thank Dr. Zhixin Shi, Dr. Srirangaraj Setlur, Prof. Venu Govindaraju (State University of New York), Dr. Ismail Haritaoglu, Dr. Gulcin Caner Harmanci (Polar Rain Inc.), Dr. Wael Abd-Almageed, and Dr. David Doermann (University of Maryland) for providing the implementations of their text line segmentation algorithms.

DISCLAIMER: The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

[1] B. Gatos, N. Stamatopoulos, and G. Louloudis, "ICDAR 2009 Handwriting Segmentation Contest," in *Proceedings of the International Conference on Document Analysis and Recognition*, 2009, pp. 1393–1397.

[2] M. Arivazhagan, H. Srinivasan, and S. N. Srihari, "A Statistical Approach to Handwritten Line Segmentation," in *Proceedings of SPIE Document Recognition and Retrieval XIV*, 2007, pp. 6500T–1–11.

[3] Y. Li, Y. Zheng, D. Doermann, and S. Jaeger, "A New Algorithm for Detecting Text Line in Handwritten Documents," in *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, 2006, pp. 35–40.

[4] S. Nicolas, T. Paquet, and L. Heutte, "Text Line Segmentation in Handwritten Document using a Production System," in *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, 2004, pp. 245–250.

[5] G. Louloudis, B. Gatos, I. Pratikakis, and K. Halatsis, "A Block-based Hough Transform Mapping for Text Line Detection in Handwritten Documents," in *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, 2006, pp. 515–520.

[6] J. Kumar, W. Abd-Almageed, L. Kang, and D. Doermann, "Handwritten Arabic Text Line Segmentation using Affinity Propagation," in *Proceedings of the International Workshop on Document Analysis Systems*, 2010, pp. 135–142.

[7] Z. Razak, K. Zulkiflee, M. Y. I. Idris, E. M. Tamil, M. Noorzaily, M. Noor, R. Salleh, M. Yaakob, and M. Yaacob, "Off-line Handwriting Text Line Segmentation : A Review," *International Journal of Computer Science and Network Security*, vol. 8, no. 7, pp. 12–20, 2008.

[8] J. G. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer Output Voting Error Reduction (ROVER)," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 1997, pp. 347–354.

[9] X. Ye, M. Cheriet, and C. Suen, "StrCombo: Combination of String Recognizers," *Pattern Recognition Letters*, vol. 23, no. 4, pp. 381–394, 2002.

[10] W. Wang, A. Brakensiek, and G. Rigoll, "Combination of Multiple Classifiers for Handwritten Word Recognition," in *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, 2002, pp. 117–122.

[11] R. Bertolami and H. Bunke, "Hidden Markov Model-based Ensemble Methods for Offline Handwritten Text Line Recognition," *Pattern Recognition*, vol. 41, no. 11, pp. 3452–3460, 2008.

[12] R. Prasad, S. Saleem, E. MacRostie, P. Natarajan, and M. Decerbo, "Multi-frame Combination for Robust Videotext Recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2008, pp. 1357–1360.

[13] Z. Shi, S. Setlur, and V. Govindaraju, "A Steerable Directional Local Profile Technique for Extraction of Handwritten Arabic Text Lines," in *Proceedings of the International Conference on Document Analysis and Recognition*, 2009, pp. 176–180.

[14] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, "A survey for the quadratic assignment problem," *European Jnl. Operations Research*, vol. 176, pp. 657–690, 2007.

[15] M. Charikar, V. Guruswami, and A. Wirth, "Clustering with qualitative information," in *FOCS '03: Proc. Symp. Foundations of Computer Science*, 2003.

[16] S. Vitaladevuni and R. Basri, "Co-clustering of image segments using convex optimization applied to em neuronal reconstruction," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010.

[17] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

[18] S. Yu and J. Shi, "Perceiving shapes through region and boundary interaction," Robotics Institute, Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-RI-TR-01-21, July 2001.